Generalized Formalization of Games

Christiaan van de Sande Tanner Reese

October 24, 2020

1 Introduction

2 Games

3 Rules

3.1 Motivation and Definition

The purpose of a rule is to define legal transitions between states in a game. It may be tempting to define a rule as a function from one state to another state, but this approach is very limited, as it does not help to describe the moves themselves, only the consequences of those moves. The definition of a rule must include the definition of a function λ that generates the legal moves from a given state and the definition of a function ϕ that executes a given move on a given state. The moves must be independent of the state, so that the same move can be executed on different states. Moves must also be reversable, meaning that, given any states s and s' and move m, where m is the move from s to s', there must be a way to find s from only s' and m. This reverse operation is the responsibility of a third function ρ. The requirements of these three functions gives the following definition for a rule:

Definition 3.1.1. r is a **rule** on sets A and B (written $r \in \mathcal{R}(A, B)$) iff $\lambda_r : A \mapsto \mathcal{P}(B)$, $\phi_r : A \times B \mapsto A$, $\rho_r : A \times B \mapsto A$, such that $\phi_r(\rho_r(a, b), b) = a$ and $\rho_r(\phi_r(a, b), b) = a$, for all $a \in A$, $b \in \lambda_r(a)$.

For the rule-of-play, r, of a game, the set of all well-formed states, S, forms the input set (above denoted as A) for r and the set of all well-formed moves, M, forms the output set (above denoted as B) for r, so $r \in \mathcal{R}(S, M)$. However, not all rules have the set of well-formed states as their input set. In fact, most do not. For most rules the input set is the state with some data added or removed based on related moves. Take, for example, the rule for capturing in chess. In most cases, captures are performed by moving one piece onto a square occupied by another piece (the notable exception to this is en-passant capturing) Thus, the legal captures are dependent, not only on the state (i.e. where all the pieces are located), but also on the move being made, a piece only captures on the square that it moves to.

It is *possible* to define a function that finds all legal captures and a second function that finds all legal moves that are not captures, but this approach sacrifices the flexibility afforded by having one rule that is responsible for the *movement* of pieces and having another rule that is responsible

for the conditions under which pieces are *captured*. In the latter case, the idea of capturing on the square to which you move is separated from the idea of moving in certain ways. That way chess movement patterns can be applied to games without capturing or with different systems for capturing and the same capturing rule applies to all of the pieces in chess (except for pawns).

3.2 Properties of rules

In order to discuss rules, it is necessary to establish some vocabulary.

Definition 3.2.1. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(B, C)$. Then s is **dependent** on r

Dependence is useful for describing which rules can be applied after others. For example, chess pieces have one rule that defines how they move, and another rule that describes what captures occur given a particular move. Thus the capturing rule is dependent on the movement rule, because which captures are possible depends on which move is being made.

Definition 3.2.2. Let $r \in \mathcal{R}(A, B)$. r is **repeatable** iff $A \supseteq B$.

Repeatable rules are rules where the output set is contained in the input set. This is means that repeatable rules can be applied again on their own outputs. This property is useful for rules that are applied in patterns, e.g. the movments of bishops and rooks in chess. Notice that repeatable rules both succeed and precede themeselves.

Definition 3.2.3. Let $r \in \mathcal{R}(A, A)$. r is **constant** iff $\lambda_r(a) = \{a\}$ for all $a \in A$.

Constant rules are rules that do not present the player with any choice as to how the rule is applied. Thus the legal moves from a constant rule on any given input is the singleton set containing only that input. Notice that all constant rules are also repeatable. One example of a constant rule is the turn mechanic common to most games. Regardless of the choices made by the players, on every turn, the player whose turn it is changes.

Definition 3.2.4. Let $r \in \mathcal{R}(A, B)$. r is **passive** iff $\phi_r(a, b) = a$ for all $a \in A$, $b \in B$.

Notice that passive rules are really boring.

Definition 3.2.5. Let $r \in \mathcal{R}(A, A)$. r is **simple** iff $\phi_r(a_1, a_2) = a_2$ for all $a_1, a_2 \in A$.

Notice that all simple rules are also repeatable. Simple rules are useful for detecting certain conditions on the board.

Definition 3.2.6. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(A, C)$. r and s are **independent** ($r \perp s$) iff, $\phi_r(\phi_s(a, c), b) = \phi_s(\phi_r(a, b), c)$, for all $a \in A$, $b \in B$, $c \in C$.

3.3 Elementary Rules

Definition 3.3.1. Let r be the **void rule** (written $r = \mathcal{R}_\varnothing$). Then $r \in \mathcal{R}(A, B)$, where:

$$\lambda_r(a) = \varnothing \tag{1}$$
$$\phi_r(a, b) = a \tag{2}$$
$$\rho_r(a, b) = a. \tag{3}$$

Notice that the void rule is passive and constant.

Definition 3.3.2. Let r be the **pass rule** (written $r = \mathcal{R}_0$). Then $r \in \mathcal{R}(A, A)$ where:

$$\lambda_r(a) = \{a\} \tag{4}$$
$$\phi_r(a_1, a_2) = a_2 \tag{5}$$
$$\rho_r(a_1, a_2) = a_2. \tag{6}$$

Notice that the pass rule is simple and constant.

Definition 3.3.3. Let $f : A \mapsto B$ be a bijection. Let r be the **rule form** of f (written $r = \mathcal{R}_f$). Then $r \in \mathcal{R}(A, B)$ where:

$$\lambda_r(a) = \{f(a)\} \tag{7}$$
$$\phi_r(a, b) = f^{-1}(b) \tag{8}$$
$$\rho_r(a, b) = f^{-1}(b). \tag{9}$$

3.4 Operations on Rules

While the rule-of-play for a game can be created using a single rule that handles all off the intricacies of the game, The advantage offered by the rule structure is the ability to combine simpler rules, each of which control only a small part of the game. Simple rules can be combined or modified into more complex ones using the following operations.

Definition 3.4.1. Let $r \in \mathcal{R}(A, B)$ and s be the **complement** of r (written $s = \overline{r}$). Then $s \in \mathcal{R}(A, A)$, where:

$$\lambda_s(a) = \begin{cases} \varnothing & \lambda_r(a) \neq \varnothing \\ \{a\} & \lambda_r(a) = \varnothing \end{cases} \tag{10}$$
$$\phi_s(a_1, a_2) = a_2 \tag{11}$$
$$\rho_s(a_1, a_2) = a_2. \tag{12}$$

Definition 3.4.2. Let $r \in \mathcal{R}(A, B)$ and s be the **reduction** of r (written $s = \overline{\overline{r}}$). Then $s \in \mathcal{R}(A, A)$, where:

$$\lambda_s(a) = \begin{cases} \varnothing & \lambda_r(a) = \varnothing \\ \{a\} & \lambda_r(a) \neq \varnothing \end{cases} \tag{13}$$
$$\phi_s(a_1, a_2) = a_2 \tag{14}$$
$$\rho_s(a_1, b_2) = a_2. \tag{15}$$

Notice that the reduction of r is equivalent to the complement of the complement of r and that both the reduction of r and the complement of r are both simple and constant for any $r \in \mathcal{R}(A, B)$.

Definition 3.4.3. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(A, C)$ and $\phi_s(a, b) = \phi_r(a, b)$ for all $a \in A, b \in B \cap C$ and t be the **sum** of s and r (written $t = s + r$). Then $t \in \mathcal{R}(A, B \cup C)$, where:

$$\lambda_t(a) = \lambda_r(a) \cup \lambda_s(a) \tag{16}$$

$$\phi_t(a, b) = \begin{cases} \phi_s(a, b) & b \in C \setminus B \\ \phi_r(a, b) & b \in B \setminus C \\ \phi_s(a, b) = \phi_r(a, b) & b \in B \cap C \end{cases} \tag{17}$$

$$\rho_t(a, b) = \begin{cases} \rho_s(a, b) & b \in C \setminus B \\ \rho_r(a, b) & b \in B \setminus C \\ \rho_s(a, b) = \rho_r(a, b) & b \in B \cap C. \end{cases} \tag{18}$$

Definition 3.4.4. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(A, C)$ and $\phi_s(a, b) = \phi_r(a, b)$ for all $a \in A, b \in B \cap C$ and t be the **intersection** of s and r (written $t = s \wedge r$). Then $t \in \mathcal{R}(A, B \cap C)$, where:

$$\lambda_t(a) = \lambda_r(a) \cap \lambda_s(a) \tag{19}$$

$$\phi_t(a, b) = \phi_s(a, b) = \phi_r(a, b) \tag{20}$$

$$\rho_t(a, b) = \rho_s(a, b) = \rho_r(a, b). \tag{21}$$

Definition 3.4.5. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(A, C)$ and t be the **independent product** of s and r (written $t = s \cdot r$). Then $t \in \mathcal{R}(A, B \times C)$, where:

$$\lambda_t(a) = \lambda_r(a) \times \lambda_s(a) \tag{22}$$

$$\phi_t(a, (b, c)) = \phi_r(\phi_s(a, c), b) \tag{23}$$

$$\rho_t(a, (b, c)) = \rho_s(\rho_r(a, b), c). \tag{24}$$

Definition 3.4.6. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(B, C)$ and t be the **dependent product** of s and r (written $t = s \circ r$). Then $t \in \mathcal{R}(A, B \times C)$, where:

$$\lambda_t(a) = \{(b, c) | b \in \lambda_r(a), c \in \lambda_s(b)\} \tag{25}$$

$$\phi_t(a, (b, c)) = \phi_r(a, \phi_s(b, c)) \tag{26}$$

$$\rho_t(a, (b, c)) = \rho_r(a, \phi_s(b, c)). \tag{27}$$

Definition 3.4.7. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(A \times B, C)$ and t be the **dependent product** of s and r (written $t = s \circ r$). Then $t \in \mathcal{R}(A, B \times C)$, where:

$$\lambda_t(a) = \{(b, c) | b \in \lambda_r(a), c \in \lambda_s((a, b))\} \tag{28}$$

$$\phi_t(a, (b, c)) = \phi_r(\phi_s((a, b), c)) \tag{29}$$

$$\rho_t(a, (b, c)) = \rho_r(\rho_s((a, b), c)). \tag{30}$$

Definition 3.4.8. Let $r \in \mathcal{R}(A, B)$ and $s \in \mathcal{R}(C, D)$ and t be the **full product** of s and r (written $t = s \times r$). Then $t \in \mathcal{R}(A \times B, C \times D)$, where:

$$\lambda_t((a, c)) = \{(b, d) | b \in \lambda_r(a), d \in \lambda_s(c)\} \tag{31}$$

$$\phi_t((a, c), (b, d)) = (\phi_r(a, b), \phi_s(c, d)) \tag{32}$$

$$\rho_t((a, c), (b, d)) = (\rho_r(a, b), \rho_s(c, d)). \tag{33}$$

Definition 3.4.9. Let $r \in \mathcal{R}(B, B)$ and $s \in \mathcal{R}(A, B)$ and $t \in \mathcal{R}(B, C)$ and v be r **patterned** from s to t (written $v = r|_s^t$). Then, $v \in \mathcal{R}(A, B)$, where:

$$\kappa_v(b) = \begin{cases} \{b\} & \lambda_t(b) \neq \varnothing \\ \{b\} \cup (\widehat{\kappa}_v \circ \lambda_r(b)) & \lambda_t(b) = \varnothing \end{cases} \tag{34}$$

$$\lambda_v(a) = \widehat{\kappa}_v \circ \lambda_s(a) \tag{35}$$

$$\phi_v(a, b) = \phi_s(a, \phi_r(b, b)) \tag{36}$$

$$\rho_v(a, b) = \rho_s(a, \rho_r(b, b)). \tag{37}$$

Definition 3.4.10. Let $n, N \in \mathbb{N}$ such that $0 < n \leq N$. Let A_i, B_i be sets for all $i \leq N$ such that $A_i = B_i$ for all $i \neq n$. Let $r \in \mathcal{R}(A_n, B_n)$. Let $s_i = \begin{cases} \mathcal{R}_0 & i \neq n \\ r & i = n \end{cases}$. Let $t = \operatorname{imp}_n^N r$. Then $t = \bigtimes_{i=1}^N s_i$, so $t \in \mathcal{R}(\bigtimes_{i=1}^N A_i, \bigtimes_{i=1}^N B_i)$.

4 Evaluators

5 Conclusion